Table of Contents

Introduction

Theoretical investigations of *switching circuits* go back to the early papers by Shannon and Lupanov [Sha38, RS42, Sha49, Lup58]. Shannon (and his co-author Riordan) used Boolean algebra to design and analyze switching circuits. Lupanov was the head of a group of Russian mathematicians, working on the question of how many gates a switching circuit must have to be able to perform certain tasks.

The theory of logical design and switching circuits then developed rapidly. The direction which *complexity theoretic research on circuits* took was heavily influenced by Savage's textbook [Sav76]. Savage (also in earlier papers) related Boolean circuits to other well-established computation models, most prominently the Turing machine. This offered the hope that the $\mathrm{P} \stackrel{?}{=} \mathrm{NP}$-problem might be solved via circuit complexity. This hope was boosted by the development of strong lower bound techniques in the 1980s.

Around that time, a variety of concrete and abstract models for parallel computers also emerged. Boolean circuits could be related to these models (in particular, the *parallel random access machine*, PRAM), facilitating the study of the computational power of such devices.

Thus the field of circuit complexity has attracted interest from a number of different viewpoints, both practical and theoretical. The reader will see throughout this book that a lot of the questions that were asked, and a lot of the definitions and constructions made, can only be understood with this two-fold influence in mind.

In Chap. 1 we start by examining a number of problems of highly practical relevance, such as basic arithmetic operations (addition, multiplication, division), counting, sorting, etc., from a circuit complexity viewpoint. We formally develop the computation model of Boolean circuits and construct efficient circuits for these problems.

The computation model of circuits has one characteristic which makes it different from other familiar models such as the random access machine or the Turing machine: A Boolean circuit has a fixed number of input gates. Thus a single circuit can only work on inputs of one fixed length (in contrast to a machine program which works for inputs of arbitrary length). If we want to solve the problems above (which are defined for different input lengths) by circuits, we therefore have to construct one circuit for each input length. This is why the Boolean circuit model is referred to as a *non-uniform* computation model, in contrast to, e. g., the Turing machine, which is a *uniform* model. The name stems from the fact that the various circuits for different input lengths may have a completely dissimilar structure. This has the consequence, as we will see, that there are even non-computable functions which can be "computed" by small circuits.

From an algorithmic point of view, this is certainly not what we wish: An algorithm for a problem should be a finite object (as a finite text in a programming language, or a finite transition function of a Turing machine). Hence an infinite family of circuits, one for each input length, will not be considered an algorithm. However, the examples we construct will have the property that the distinct circuits in the family have a very regular structure, in particular, circuits for different input lengths do not differ too much. This makes it possible to give a finite description of the structure of the infinite family. If such a description is possible then we say the circuit family is *uniform*. We will examine several ways to make these ideas precise, leading to various so-called *uniformity conditions*.

In Chap. 2 we compare the model of Boolean circuits with a number of other computation models, such as deterministic and alternating Turing machines and parallel random access machines. We will show how these machines can simulate uniform circuit families, and, vice versa, how circuits can be constructed to simulate the computation of such machines. We will also consider so-called *non-uniform Turing machines* and relate them to non-uniform circuits.

In Chap. 3 we turn to the theory of lower bounds. We start with the examination of some abstract problems which, however, in the end lead to results about the complexity of the problems defined and examined in Chap. 1. We will thus be able to give very precise statements about the circuit complexity of problems like sorting, multiplication, etc.

All lower bounds given in this chapter will indeed hold for non-uniform circuits; e. g., we will show that multiplication cannot even be computed by non-uniform circuits of a certain complexity. In Chap. 4 we come back to our algorithmic standpoint and hence to the study of uniform circuits. We examine the class NC, a class of immense importance in the theory of parallel algorithms. This class, in a sense, captures the notion of problems with very fast parallel algorithms using a feasible amount of hardware. We will consider different aspects of the class NC and its structure, and come up with relations to diverse fields of mathematics such as group theory or finite model theory. A further computation model, the so-called *branching programs*, will turn out to be important in this chapter.

In Chap. 4 we also develop a theory of P-completeness, analogous to the well-known NP-completeness theory. P-complete problems are the "hardest" problems in the class P, and they will most likely not be in NC, i. e., they will not have efficient parallel algorithms. We will encounter a large number of examples in this book.

In Chap. 5 we turn to circuits whose basic components are not logical gates but addition and multiplication gates. This study is thus not immediately motivated from a VLSI or engineering point of view, as is the case for Boolean circuits. The importance of this model stems from the fact that it serves as a theoretical basis for the field of computer algebra. There are interesting connections between Boolean circuits and these so-called *arithmetic circuits,* as we will see.

All problems considered in Chaps. 1–5 have efficient sequential algorithms, i. e., they are members of the class P. In Chap. 6 we turn to higher classes and show how they relate to Boolean circuits. We also come back to the theory of lower bounds in this final chapter, and we examine the power of (non-uniform) small circuits, where "small" means having a number of gates bounded by a polynomial in the input length. Can, for example, NP-complete problems be solved by such circuits? We will also present lower bounds which are known to hold only for uniform circuits. We will, for instance, show that certain functions cannot be computed by so-called uniform *threshold circuits* of small size, but it is still open whether small non-uniform threshold circuits exist for these problems.